UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
10/644,399	08/19/2003	Francis X. McKeen	42P15739	7924

8791 7590 07/09/2008
BLAKELY SOKOLOFF TAYLOR & ZAFMAN LLP
1279 OAKMEAD PARKWAY
SUNNYVALE, CA 94085-4040

EXAMINER	
DOLLINGER, TONIA LYNN MEONSKE	
ART UNIT	PAPER NUMBER
2181	
NOTIFICATION DATE	DELIVERY MODE
07/09/2008	ELECTRONIC

Please find below and/or attached an Office communication concerning this application or proceeding.

The time period for reply, if any, is set in the attached communication.

Notice of the Office communication was sent electronically on above-indicated "Notification Date" to the following e-mail address(es):

Office Action Summary	Application No.	Applicant(s)
	10/644,399	MCKEEN, FRANCIS X.
	Examiner	Art Unit
	Tonia LM Dollinger	2181

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE _3_ MONTH(S) OR THIRTY (30) DAYS, WHICHEVER IS LONGER, FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

1)☒ Responsive to communication(s) filed on _08 April 2008_.

2a)☐ This action is **FINAL**. 2b)☒ This action is non-final.

3)☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

4)☒ Claim(s) _1,5-7,9 and 11-19_ is/are pending in the application.

4a) Of the above claim(s) _____ is/are withdrawn from consideration.

5)☐ Claim(s) _____ is/are allowed.

6)☒ Claim(s) _1,5-7,9,11-19_ is/are rejected.

7)☐ Claim(s) _____ is/are objected to.

8)☐ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

9)☒ The specification is objected to by the Examiner.

10)☐ The drawing(s) filed on _____ is/are: a)☐ accepted or b)☐ objected to by the Examiner.

Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).

11)☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. § 119

12)☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

a)☐ All b)☐ Some * c)☐ None of:

1.☐ Certified copies of the priority documents have been received.

2.☐ Certified copies of the priority documents have been received in Application No. _____.

3.☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

Attachment(s)

1) ☐ Notice of References Cited (PTO-892)

2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)

3) ☐ Information Disclosure Statement(s) (PTO/SB/08) Paper No(s)/Mail Date _____.

4) ☐ Interview Summary (PTO-413) Paper No(s)/Mail Date. _____ .

5) ☐ Notice of Informal Patent Application

6) ☐ Other: _____.

DETAILED ACTION

Specification

1. Claims 15 and 16 are objected to because of the following informalities:

   a. The specification is objected to as failing to provide proper antecedent basis for the claimed subject matter. See 37 CFR 1.75(d)(1) and MPEP § 608.01(o). Correction of the following is required: In claims 15-16, line 1, the limitation "computer readable medium" is not described in the specification. Appropriate correction is required.

Claim Objections

2. Claims 9 and 19 are objected to because of the following informalities: The following limitation appearing in both claims is redundant "wherein the exception handler determines if the return address from the first stack or the return address from the second stack is to be used as a value for an instruction pointer, wherein the exception handler determines if the return address from the first stack, or the return address from the second stack is to be used as a value for an instruction pointer". Please delete the second wherein clause since it is essentially the same as the first wherein clause. Appropriate correction is required.

Claim Rejections - 35 USC § 101

3. 35 U.S.C. 101 reads as follows:

Whoever invents or discovers any new and useful process, machine, manufacture, or composition of matter, or any new and useful improvement thereof, may obtain a patent therefor, subject to the conditions and requirements of this title.

4. Claims 15 and 16 are rejected under 35 U.S.C. 101 because the claimed invention is directed to non-statutory subject matter. In claims 15-20, line 1, the limitation "computer readable medium" is not supported in the specification. With the limitation it appears that applicant may be referring to signal bearing media on page 9, line 14, of the specification. If so then transmission type media (which is not tangible) is included in the limitation (see page 9, line 13) and the claim is not patentable. Examiner suggests changing "computer readable medium" to "computer recordable type media" to be consistent with the specification on page 9, line 15 and to avoid this 101 rejection. Appropriate correction is required.

Claim Rejections - 35 USC § 102

5. The following is a quotation of the appropriate paragraphs of 35 U.S.C. 102 that form the basis for the rejections under this section made in this Office action:

> A person shall be entitled to a patent unless –
>
> (e) the invention was described in (1) an application for patent, published under section 122(b), by another filed in the United States before the invention by the applicant for patent or (2) a patent granted on an application for patent by another filed in the United States before the invention by the applicant for patent, except that an international application filed under the treaty defined in section 351(a) shall have the effects for purposes of this subsection of an application filed in the United States only if the international application designated the United States and was published under Article 21(2) of such treaty in the English language.

6. Claims 1, 5-7, 9, 11-19 are rejected under 35 U.S.C. 102(e) as being anticipated by Lee et al., US Patent 6,996,677 (herein after referred to as Lee).

7. Referring to claim 1, Lee has taught a method, comprising:

a. encountering a function call instruction that calls a called function during program execution (abstract, column 2, lines 29-50, jump routine);

b. saving a return address in a first stack and in a second stack at the same time, the return address containing an instruction to be executed after execution of the called function (abstract, lines 1-12, column 2, lines 29-50, A return address is saved in a first stack and a second stack upon encountering a jump routine. The first stack is element 118 in Figure 2. The second stack is shown in Figure 2, elements 110a-110n. 110a-110n is a computer memory consisting of arrays of memory elements stacked one on top of another, which is a stack. See Merriam-webster's online dictionary to support this definition of stack.), wherein the second stack stores a return address for each function call instruction that calls a called function during program execution (column 4, lines 6-17, When a subroutine is called during program execution, the return address is stored in element 110a-n.);

c. executing the called function (abstract, column 2, lines 29-50, A jump to subroutine is executed.);

d. determining if the return address stored in the first stack matches the return address stored in the second stack to provide protection from a buffer overflow attack (abstract, column 2, lines 29-50, first comparator and second comparator);

e. executing exception handling code if an exception is generated when the return addresses do not match (abstract, column 2, lines 29-50, An interrupt signal is generated if the addresses are not the same. Exception handling software is continually executed by the protection co-processor.),

f. wherein the exception handling code determines what value to pass to a program pointer based on the return address retrieved from each of the first and second stacks (abstract, column 2, lines 29-50, column 4, line 38-column 5, line 18. column 5, lines 38-41, column 6, lines 59-column 7, line 40, When the values retrieved from both stacks are equal, then the program counter is updated with the current return address value, otherwise an exception is generated to pass the correct value to the program counter.).

8. Referring to claim 5, Lee has taught the method of claim 1, as described above, and wherein the exception handling code terminates execution of the program (abstract, column 2, lines 29-50, column 4, line 38-column 5, line 18. column 5, lines 38-41, column 6, lines 59-column 7, line 40, The instruction to move data to the PC register is aborted.).

9. Referring to claim 6, Lee has taught a method, comprising:

a. processing instructions within a virtual machine (abstract, column 2, lines 29-50, column 5, lines 39-41, A software jump/return routine is executed.);

b. saving a return address in a first stack and in a second stack at the same time, the return address being an address at which program execution is to resume after execution of a called function (abstract, column 2, lines 29-50);

c. comparing the return addresses saved in the first and second stack upon execution of the called function (abstract, column 2, lines 29-50, first comparator); and

d. exiting the virtual machine if the return addresses do not match to provide protection from a buffer overflow attack (abstract, column 2, lines 29-50, column 4, line 38-column 5, line 18. column 5, lines 38-41, column 6, lines 59-column 7, line 40, The instruction to move data to the PC register is aborted.) wherein the second stack stores a return address for each function call instruction that calls a called function during program execution (column 4, lines 6-17, When a subroutine is called during program execution, the return address is stored in element 110a-n.), and wherein an exception handler determines if the return address from the first stack or the return address from the second stack is to be used as a value for an instruction pointer (abstract, column 2, lines 29-50, column 4, line 38-column 5, line 18. column 5, lines 38-41, column 6, lines 59-column 7, line 40, When the return values retrieved from both stacks are equal, then the program counter is updated with the current return address value, otherwise an exception is generated to pass the correct value to the program counter.).

10. Referring to claim 7, Lee has taught the method of claim 6, as described above, and further comprising passing control to the exception handler (abstract, column 2, lines 29-50, column 4, line 38-column 5, line 18, column 5, lines 38-41, column 6, lines 59-column 7, line 40, An interrupt signal is generated if the addresses are not the same to load the PC register with a correct value.).

11. Referring to claim 9, Lee has taught a method, comprising:

a. creating first and second stacks for a program during execution of the program (abstract, column 2, lines 29-50, column 4, line 38-column 5, line 18, Values for the first and second stacks are created and pushed on the stacks during program execution.);

b. encountering a function call to a called function (abstract, column 2, lines 29-50, column 4, line 38-column 5, line 18, jump to subroutine);

c. storing data for the called function and a return address in the first stack (abstract, column 2, line 29-column 3, line 15, column 3, line 54-column 4, lines 26, column 4, line 38-column 5, line 18, first stack);

d. storing the return address in the second stack at the same time as the first stack (abstract, column 2, line 29-column 3, line 15, column 3, line 54-column 4, lines 26, column 4, line 38-column 5, line 18, second stack); and

e. passing control of the program to an exception handler if the return address stored in the first stack does not match the return address stored in the second stack upon execution of the called function to provide protection from a buffer overflow attack (abstract, column 2, line 29-column 3, line 15, column 3, line 54-column 4, lines 26, column 4, line 38-column 5, line 18, When the addresses do not match an exception is generated.), wherein the exception handler determines if the return address from the first stack or the return address from the second stack is to be used as a value for an instruction pointer, wherein the exception handler determines if the return address from the first stack, or the return address from the second stack is to be used as a value for an instruction pointer (abstract, column 2, lines 29-50, column 4, line 38-column 5, line 18. column 5, lines 38-41, column 6, lines 59-column 7, line 40, When the return address values retrieved from both stacks are equal, then the program counter is updated with the current return address value, otherwise an exception is generated to pass the correct value to the program counter.).

12. Referring to claim 11, Lee has taught a processor, comprising:

a. memory management logic to allocate first and second memory locations corresponding to first and second stacks, respectively, when a function call instruction calls to a called function is encountered during program execution (abstract, column 2, lines 29-50, column 4, line 38-column 5, line 18, Values for the first and second stacks are created and pushed on the stacks during program execution of jump to subroutines.);

b. function call logic to write a return address to a memory location from the first memory locations and to a memory location from the second memory locations at the same time (abstract, lines 19-22, column 2, lines 29-50, A return address is saved in a first stack and a second stack upon encountering a jump routine.), the return address being an address at which program flow is to resume after execution of the called function (abstract, lines 19-22, column 2, lines 29-50, The return address is loaded into the program counter such that program flow resumes after executing a jump instruction.); and

c. buffer overflow control logic to determine if the return address retrieved from the first memory locations matches the return address retrieved from the second memory locations, upon execution of the called function to provide protection from a buffer overflow attack (abstract, column 2, lines 29-50, first comparator and second comparator) wherein the exception handler determines if the return address from the first stack, or the return address from the second stack is to be used as a value for an instruction pointer (abstract, column 2, lines 29-50, column 4, line 38-column 5, line 18. column 5, lines 38-41, column 6, lines 59-column 7, line 40, When the return address values retrieved from both stacks are equal, then the program counter is updated with the current return address value, otherwise an exception is generated to pass the correct value to the program counter.).

13. Referring to claim 12, Lee has taught the processor of claim 11, as described above, and wherein the function call logic and the buffer overflow control logic comprises microcode stored within the processor (column 5, lines 39-41).

14. Referring to claim 13, Lee has taught a system, comprising:

a. a memory (Figure 3, element 102); and

b. a processor coupled to the memory (Figure 3, at least elements 101, 140, 142, and 144), the processor comprising memory management logic to allocate first and second memory locations corresponding to first and second stacks, respectively, when a function call instruction that calls a called function is encountered during program execution (abstract, column 2, lines 29-50, column 4, line 38-column 5, line 18, Values for the first and second stacks are created and pushed on the stacks during program execution.);

c. function call logic to write a return address to a memory location from the first memory locations and to a memory location from the second memory locations at the same time (abstract, lines 19-22, column 2, lines 29-50, A return address is saved in a first stack and a second stack upon encountering a jump routine.), the return address being an address at which program flow is to resume after execution of the called function (abstract, lines 19-22, column 2, lines 29-50, The return address is loaded into the program counter such that program flow resumes after executing a jump instruction.); and

d. buffer overflow control logic to determine if the return address retrieved from the first memory locations matches the return address retrieved from the second memory locations, upon execution of the called function to provide protection from a buffer overflow attack (abstract, column 2, lines 29-50, first comparator and second comparator) wherein the exception handler determines if the return address from the first stack, or the return address from the second stack is to be used as a value for an instruction pointer (abstract, column 2, lines 29-50, column 4, line 38-column 5, line 18. column 5, lines 38-41, column 6, lines 59-column 7, line 40, When the return address values retrieved from both stacks are equal, then the program counter is updated with the current return address value, otherwise an exception is generated to pass the correct value to the program counter.).

15. Referring to claim 14, Lee has taught the system of claim 13, as described above, and wherein the memory management logic, the function call logic, and the

buffer overflow control logic comprise microcode stored within the processor (column 5, lines 39-41).

16. Referring to claim 15, Lee has taught a computer readable medium having stored thereon a sequence of instructions which when executed by a processor, cause the processor to perform a method comprising:

a. encountering a function call instruction that calls a called function during program execution (abstract, column 2, lines 29-50, jump routine);

b. saving a return address in a first stack and in a second stack at the same time (abstract, lines 19-22, column 2, lines 29-50, A return address is saved in a first stack and a second stack upon encountering a jump routine.), the return address containing an instruction to be executed after execution of the called function (abstract, lines 19-22, column 2, lines 29-50, The return address is loaded into the program counter such that program flow resumes after executing a jump instruction.);

c. executing the called function (abstract, column 2, lines 29-50, A jump to subroutine is executed.); and

d. determining if the return address stored in the first stack matches the return address stored in the second stack to provide protection from a buffer overflow attack (abstract, column 2, lines 29-50, first comparator and second comparator) wherein the exception handler determines if the return address from the first stack, or the return address from the second stack is to be used as a value for an instruction pointer (abstract, column 2, lines 29-50, column 4, line

38-column 5, line 18. column 5, lines 38-41, column 6, lines 59-column 7, line 40, When the return values retrieved from both stacks are equal, then the program counter is updated with the current return address value, otherwise an exception is generated to pass the correct value to the program counter.).

17. Referring to claim 16, Lee has taught the computer readable medium of claim 15, as described above, and wherein the method further comprises generating an exception if the return addresses do not match (abstract, column 2, lines 29-50, An interrupt signal is generated if the addresses are not the same.).

18. Referring to claim 17, Lee has taught a computer recordable type medium having stored thereon a sequence of instructions which when executed by a processor, cause the processor to perform a method comprising:

a. processing instructions within a virtual machine (abstract, column 2, lines 29-50, column 5, lines 39-41, A software jump/return routine is executed.);

b. saving a return address in a first stack and in a second stack at the same time (abstract, lines 19-22, column 2, lines 29-50, A return address is saved in a first stack and a second stack upon encountering a jump routine.), the return address being an address at which program execution is to resume after execution of a called function (abstract, lines 19-22, column 2, lines 29-50, The return address is loaded into the program counter such that program flow resumes after executing a jump instruction.);

c. comparing the return addresses saved in the first and second stack upon execution of the called function (abstract, column 2, lines 29-50, first comparator and second comparator); and

d. exiting the virtual machine if the return addresses do not match to provide protection from a buffer overflow attack (abstract, column 2, lines 29-50, column 4, line 38-column 5, line 18. column 5, lines 38-41, column 6, lines 59-column 7, line 40, The instruction to move data to the PC register is aborted.) wherein the exception handler determines if the return address from the first stack, or the return address from the second stack is to be used as a value for an instruction pointer (abstract, column 2, lines 29-50, column 4, line 38-column 5, line 18. column 5, lines 38-41, column 6, lines 59-column 7, line 40, When the return values retrieved from both stacks are equal, then the program counter is updated with the current return address value, otherwise an exception is generated to pass the correct value to the program counter.).

19. Referring to claim 18, Lee has taught the computer recordable type medium of claim 17, as described above, and wherein the method further comprises passing control to an exception handler (abstract, column 2, lines 29-50, column 4, line 38-column 5, line 18, column 5, lines 38-41, column 6, lines 59-column 7, line 40, An interrupt signal is generated if the addresses are not the same to load the PC register with a correct value.).

20. Referring to claim 19, Lee has taught a computer recordable type medium having stored thereon a sequence of instructions which when executed by a processor, cause the processor to perform a method comprising:

a. creating first and second stacks for a program during execution of the program (abstract, column 2, lines 29-50, column 4, line 38-column 5, line 18, Values for the first and second stacks are created and pushed on the stacks during program execution.);

b. encountering a function call to a called function (abstract, column 2, lines 29-50, jump routine);

c. storing data for the called function and a return address in the first stack (abstract, column 2, line 29-column 3, line 15, column 3, line 54-column 4, lines 26, column 4, line 38-column 5, line 18, first stack);

d. storing the return address in the second stack at the same time as the first stack (abstract, column 2, line 29-column 3, line 15, column 3, line 54-column 4, lines 26, column 4, line 38-column 5, line 18, second stack); and

e. passing control of the program to an exception handler if the return address stored in the first stack does not match the return address stored in the second stack upon execution of the called function to provide protection from a buffer overflow attack (abstract, column 2, line 29-column 3, line 15, column 3, line 54-column 4, lines 26, column 4, line 38-column 5, line 18, When the addresses do not match an exception is generated.) wherein the exception handler determines if the return address from the first stack, or the return address from the second stack is to be used as a value for an instruction pointer, wherein the exception handler determines if the return address from the first stack and the return address from the second stack is to be used as a value for an instruction pointer (abstract, column 2, lines 29-50, column 4, line 38-column 5, line 18. column 5, lines 38-41, column 6, lines 59-column 7, line 40, When the return values retrieved from both stacks are equal, then the program counter is updated with the current return address value, otherwise an exception is generated to pass the correct value to the program counter.).

Response to Arguments

21. Applicant's arguments filed May 8, 2008 have been fully considered but they are not persuasive.

22. On page 7 Applicant argues that claims 15-20 have all been amended to correct the objection to the specification and the 35 USC 101 rejection. However Examiner notes that claims 15 and 16 have not been amended at all to correct the objection to the specification and the 35 USC 101 rejection. Therefore appropriate correction is still required. See objection and rejection above.

23. On page 8, Applicant argues in essence:

> *"Lee does not disclose or suggest storing the return address in a first stack and a second stack at the same time."*

However, before the subroutine is completed, there is some common time that the return address is stored in both the first and second stacks. On completion of the subroutine, a comparator compares the address currently stored in the first and second locations. So Lee has in fact taught storing the return address in a first stack and a second stack at the same time (abstract, lines 1-12, column 2, lines 29-50, A return address is saved in a first stack and a second stack upon encountering a jump routine.) Therefore this argument is moot.

24. On page 8, Applicant argues in essence:

"In contrast with claim 1, second and third locations, are separate from the stack memory (see abstract) Hence, Lee does not disclose or suggest the storing of the return address in the first and second stacks at the same time, where the second stack stores a return address for each function call instruction that calls a called function during program execution, as in Claim 1."

Examiner acknowledges that the second location (Figure 2, element 110) is separate from stack 118. However, the second location (Figure 2, element 110a-n) is still a stack. 110a-n is a computer memory consisting of arrays of memory elements stacked one on top of another, which is a stack. See Merriam-Webster's online dictionary to support this definition of stack. So Lee has in fact taught storing the return address in the first and second stacks at the same time (abstract, lines 1-12, column 2, lines 29-50, A return address is saved in a first stack and a second stack upon encountering a jump routine. The first stack is element 118 in Figure 2. The second stack is shown in Figure 2, elements 110a-110n.). Therefore this argument is moot.

25. On pages 8 and 9, Applicant argues in essence:

"Lee does not disclose or suggest executing an exception handling code when the return addresses do not mach, wherein the exception handing code determines what value to pass a program pointer based on the return address retrieved from each of the first and second stacks, as in Claim 1."

However, Lee has taught executing an exception handling code when the return addresses do not mach, wherein the exception handing code determines what value to pass a program pointer based on the return address retrieved from each of the first and second stacks (abstract, column 2, lines 29-50, column 4, line 38-column 5, line 18. column 5, lines 38-41, column 6, lines 59-column 7, line 40, When the values retrieved from both stacks are equal, then the program counter is updated with the current return address value, otherwise an exception is generated to pass the correct value to the program counter). The exception handling code necessarily determines what value to pass to the program pointer based on the retrieved return addresses. If the return addresses are the same, then the current value is passed, but if the return addresses are different, then the correct value must be calculated. The calculation of the correct value is necessarily based on the return values, because the return values determine whether a new correct value needs to be calculated or not. Applicant has not claimed that once the values are determined to be different then the two returned values are used in an algorithm/equation to calculate a correct value. If Applicant would like this or some other specific meaning read into the claims, then applicant is respectfully requested to claim those limitations. Therefore this argument is moot.

Conclusion

Any inquiry concerning this communication or earlier communications from the examiner should be directed to Tonia LM Dollinger whose telephone number is (571)

272-4170. The examiner can normally be reached on Monday-Friday with first Friday's off.

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Alford Kindred can be reached on (571) 272-4037. The fax phone number for the organization where this application or proceeding is assigned is 571-273-8300.

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see http://pair-direct.uspto.gov. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free). If you would like assistance from a USPTO Customer Service Representative or access to the automated information system, call 800-786-9199 (IN USA OR CANADA) or 571-272-1000.

TLMD

/Tonia LM Dollinger/
Primary Examiner, Art Unit 2181